

# Datenbanken und Datensicherheit

Prof. Dr. rer. nat. Peter Gerwinski

6. November 2024

# Datenbanken und Datensicherheit

<https://gitlab.cvh-server.de/pgerwinski/dbs>

## 1 Einführung

## 2 Kurzeinführung Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

## 3 Kurzeinführung TCP/IP

3.1 MAC-Adressen

3.2 IP-Adressen

3.3 TCP- und UDP-Ports

...

...



Änderungen  
vorbehalten

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

## 2.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

## 2.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

## 2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Befehlszeile übernehmen

```
$ find . -name "dbs-2024*.tex"  
./20241002/dbs-2024ws-p0.tex  
./20241002/dbs-20241002.tex  
./20241030/dbs-20241030.tex  
./20241016/dbs-20241016.tex  
./20241009/dbs-20241009.tex  
./20241106/dbs-20241106.tex  
$ grep -l "Sägefisch" $(find . -name "dbs-2024*.tex")  
./20241030/dbs-20241030.tex  
./20241030/dbs-20241106.tex
```

Alternative Schreibweise: ``...`` statt `$(...)`

## 2.5 Pipes

Standard-Ausgabe von Programm A  
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

## 2.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```



## 2.5 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v logo
```

```
es-20191002.pdf
```

```
Zeichen_123.pdf
```

```
$ ls -l $(ls *.pdf | grep -v logo)
```

```
-rw-r--r-- 1 ... 4619138 Okt 8 21:28 es-20191002.pdf
```

```
lrwxrwxrwx 1 ...          25 Okt 3  2016 Zeichen_123.pdf -> ...
```

## 2.6 Verzweigungen und Schleifen

```
$ if grep Blubb test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

## 3 Kurzeinführung TCP/IP

### 3.1 MAC-Adressen

MAC = Media Access Control

MAC-Adresse = Hardware-Adresse = Ethernet-Adresse

- `ip neigh`  
`arp`

## 3.2 IP-Adressen

- `ip addr` (Linux)  
  `ifconfig` (Unix allgemein)  
  `ipconfig` (MS Windows)
- `ip addr add <Netz>`
- `ip link`
- `ping <IP-Adresse>`

```
# ifconfig
```

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    [...]
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.42.101  netmask 255.255.255.0
                                broadcast 192.168.42.255
    ether be:3f:ca:aa:7e:51 txqueuelen 1000  (Ethernet)
    [...]
```

## 3.2 IP-Adressen

IPv4-Adressen:

- 32 Bit
- dezimal, 4 Gruppen zu je 8 Bit (0–255), durch Punkte getrennt

IPv6-Adressen:

- 128 Bit
- hexadezimal, 8 Gruppen zu je 4 Hex-Ziffern, durch Doppelpunkte getrennt
- Führende Nullen dürfen weggelassen werden.
- Zwei Doppelpunkte bedeuten: Mit Nullen auffüllen.
- Literatur und Beispiel: <https://de.wikipedia.org/wiki/IPv6>

## 3.3 TCP- und UDP-Ports

- `nc <IP> <Port>`  
Verbindung zu Programm <Port> auf Rechner <IP> aufnehmen
- `nc -l <Port>` oder `nc -p <Port> -l`  
auf eingehende Verbindungen warten („lauschen“)
- `nc -c <Programm>`  
Programm statt Mensch kommunizieren lassen
- TCP-Ports: Verbindungskonzept, Netzwerk prüft
- UDP-Ports: einzelne Pakete, Anwendung muß selbst prüfen
- ICMP: keine Ports, nur Rechner:  
Erreichbarkeit, Eigenschaften der Übertragung

Anwendung: HTTP, SMTP, ...
Transport: TCP-/UDP-Ports, ICMP
Internet: IP-Adresse
Netzwerkzugang: Hardware-/MAC-Adresse

## 3.4 TCP-Protokolle

- **HTTP**

GET / HTTP/1.1

Host: www.hs-bochum.de

(Leerzeile)

## 3.4 TCP-Protokolle

- **HTTP**

```
GET / HTTP/1.1
```

```
Host: www.hs-bochum.de
```

```
(Leerzeile)
```

- **SMTP**

```
HELO cassini
```

```
MAIL FROM: <example@example.com>
```

```
RCPT TO: <beispiel@example.de>
```

```
(E-Mail-Header – Teil der Nutzdaten)
```

```
From: Eddie Example <example@example.com>
```

```
To: Bert Beispiel <beispiel@example.de>
```

```
Subject: Hello, world!
```

```
(Leerzeile)
```

```
Hi, there!
```

```
.
```



## 3.4 TCP-Protokolle

- **HTTP**

```
GET / HTTP/1.1  
Host: www.hs-bochum.de  
(Leerzeile)
```

- **SMTP**

```
HELO cassini  
MAIL FROM: <example@example.com>  
RCPT TO: <beispiel@example.de>  
(E-Mail-Header – Teil der Nutzdaten)  
From: Eddie Example <example@example.com>  
To: Bert Beispiel <beispiel@example.de>  
Subject: Hello, world!  
(Leerzeile)  
Hi, there!  
.
```

- Protokolle „mal eben“ selbst schreiben: `nc -c` oder `inetd`

## 3.5 Routing

- `ip route` (Linux)  
`route` (MS-Windows, Unix)  
`netstat -nr` (MacOS)

```
# route -n
```

```
Kernel-IP-Routentabelle
```

Ziel	Router	Genmask	[...]	Iface
0.0.0.0	192.168.42.1	0.0.0.0	[...]	wlan0
169.254.0.0	0.0.0.0	255.255.0.0	[...]	wlan0
192.168.42.0	0.0.0.0	255.255.255.0	[...]	wlan0

Netzmaske:

Wenn nach Und-Verknüpfung mit IP-Adresse gleich, → im gleichen Netz

255.255.240.0 ist dasselbe wie /20

(20 Bit sind 1; die restlichen 12 Bit sind 0)